A Intuition for the Models

Figure 4 provides a visual comparison of HMMs, OOMs, and HQMMs, and we provide a more intuitive description of these models below:

- HMMs: These are usually parameterized by a *transition matrix* $\mathbf{A}$ where $A_{ij} = p(x_{t+1} = i | x_t = j)$ and an *emission matrix* $\mathbf{C}$, where $C_{ij} = p(o_t = i | x_t = j)$. Since these entries are conditional probabilities, all entries must be positive and the columns must sum to 1.

- OOMs: If we were to construct s new matrices, by constructing a diagonal $n \times n$ matrix from each row of $\mathbf{C}$ and multiply with $\mathbf{A}$, we would obtain the set $\{\mathbf{T}_y\}$ where each entry $(T_y)_{ij} = p(x_{t+1} = i, o_t = y | x_t = j)$. These are HMMs in the OOM representation. OOMs however are more general, as they do not require such an interpretation of each entry; as long as the operators are normalized and only produce non-negative numbers when used to compute probabilities, they are valid. This flexibility means they also cannot be given a constructive form.

- K-HQMMs: Like OOMs, these models have a tensor structure. However, they have multiple (w) operators per observable. We show the equivalence with L-HQMMs, which are specific kinds of OOMs, in the main paper.

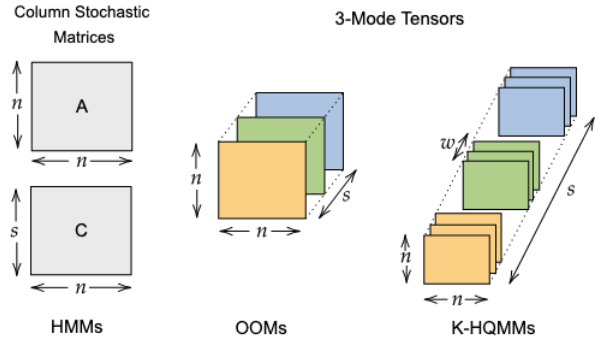


Figure 4: **Visualizing the Models**: A visualization of the matrices that parameterize the various models

B Uniqueness of L-HQMMs

While L-HQMMs (Definition 5) and and K-HQMMs (Definition 6) are equivalent representations of HQMMs, the former has the added benefit of providing a unique representation of the underlying CP map. Namely, a CP map can be equivalently defined using Kraus operator sets, which may not even have the same number of operators. This makes it difficult to directly compare two K-HQMM models, perhaps to check for equivalency. In contrast, the Liouville superoperator of a CP-map is unique, and can be canonically factorized as follows (Wood et al., 2015; Miszczak, 2011):

$$\mathbf{L} = \sum_w \mathbf{K}_w^* \otimes \mathbf{K}_w = \sum_{i=1}^{r} \gamma_i (\mathbf{K}_i^* \otimes \mathbf{K}_i) \qquad (12)$$

where $\{\mathbf{K}_w\}$ is a set of arbitrary Kraus operators, $\{\sqrt{\gamma_i}\,\mathbf{K}_i\}$ the set of *canonical* Kraus operators defining the CP map, and r the 'Kraus-rank' of the CP map. It is a well known result that these factors can be computed directly from an SVD of the Choi matrix (the 'reshuffled' Liouville matrix); the i-th singular value and vector pair correspond to γ_i and $\text{vec}(\mathbf{K}_i)$ (Wood et al., 2015; Miszczak, 2011). We illustrate this process in Figure 5.

The Kraus-rank of a CP map is equal to the rank of the Choi matrix, and is equal to the minimum number of Kraus operators required to express the operation. Since the Liouville superoperator (or the Choi matrix) uniquely defines a CP map, we can use this representations to compare two L-HQMMs. This also provides a way to compare two K-HQMMs by first converting them to their corresponding L-HQMMs.

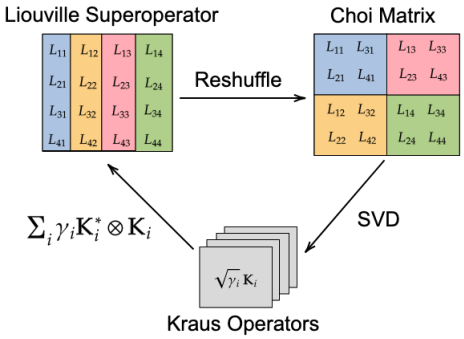


Figure 5: **Three equivalent formulations of a CP map**: The unique canonical operator sum representation of a CP map can be obtained by performing an SVD of its Choi matrix, which is obtained by reshuffling its Liouville superoperator.

C Retractions on the Stiefel Manifold

The Wen-Yin Update Scheme Given a gradient $\mathbf{G}$ of the loss function $\mathcal{L}$ with respect to parameters $\boldsymbol{\kappa}$, we wish to find the trajectory $\gamma(\tau)$ for some step size τ that corresponds to stepping along the direction of the gradient while staying on the Stiefel manifold. The Wen-Yin approach achieves this through *retractions* that smoothly map $\mathbf{G}$ or any point on a manifold's tangent bundle onto the manifold itself, while preserving the descent direction at that point (Absil et al., 2007). We can intuitively think of a retraction as wrapping the direction of $\mathbf{G}$ onto the surface of the manifold. This provides us with a feasible path $\gamma(\tau)$ for curvilinear

descent with respect to an initial feasible solution $\boldsymbol{\kappa}_0$:

$$\gamma(\tau) = \boldsymbol{\kappa}_0 - \tau \mathbf{U}\Big(\mathbb{I} + \frac{\tau}{2}\mathbf{V}^\dagger\mathbf{U}\Big)^{-1}\mathbf{V}^\dagger\boldsymbol{\kappa}_0, \tag{13}$$

where $\mathbf{U} = [\mathbf{G} \mid \boldsymbol{\kappa}_0]$, $\mathbf{V} = [\boldsymbol{\kappa}_0 \mid -\mathbf{G}]$, and $\mathbf{G}$ is the gradient at $\boldsymbol{\kappa}_0$. To see that $\gamma(\tau)$ is, in fact, the direction of steepest descent to feasibly optimize our loss, we can check if two important criteria are met. First, when $\tau = 0$, we should be at the initial point $\boldsymbol{\kappa}_0$ with $\gamma(0)$ pointing the same direction as $\mathbf{G}$. This is easily verified since $\gamma(0) = \boldsymbol{\kappa}_0$ and $\gamma'(0) = -\mathbf{G}$ (Wen and Yin, 2013). Second, any point along $\gamma(\tau)$ must be feasible. To confirm this, note that Equation 13 can be equivalently written as the following Crank-Nicolson-like update

$$\gamma(\tau) = \Big(\mathbb{I} + \frac{\tau}{2}\mathbf{A}\Big)^{-1}\Big(\mathbb{I} - \frac{\tau}{2}\mathbf{A}\Big)\,\boldsymbol{\kappa}_0, \tag{14}$$

where $\mathbf{A} = \mathbf{G}\boldsymbol{\kappa}_0^\dagger - \boldsymbol{\kappa}_0\mathbf{G}^\dagger$. Thus, $\gamma(\tau)$ is actually the Cayley transform of the skew-symmetric matrix $\mathbf{A}$ applid to $\boldsymbol{\kappa}_0$. Using this interpretation, it can be shown (Wen and Yin, 2013) that $\gamma(\tau)^\dagger\gamma(\tau) = \boldsymbol{\kappa}_0^\dagger\boldsymbol{\kappa}$. Therefore, as long as the initial point is feasible ($\boldsymbol{\kappa}_0^\dagger\boldsymbol{\kappa}_0 = \mathbb{I}$), every point along $\gamma(\tau)$ will be feasible. While Equation 14 is easier to interpret, Equation 13 is computationally favorible as it requires the inversion of a smaller $2n \times 2n$ matrix.

We combine the Wen-Yin update with a simple gradient descent scheme (Algorithm 1) to learn feasible parameters for HQMMs. In our experiments with $N = |\mathcal{O}|w$, and for a batch with m sequences of length l, we compute the loss using Equation 10 in $O(mlwn^3)$ time, perform auto-differentiation, and obtain a retraction using Equation 11 in $O(|\mathcal{O}|wn^3)$ time.

Alternative Update Schemes Algorithms that constrain parameters on the Stiefel manifold are generally either projection-like (which re-orthogonalize the naive gradient descent updates) or geodesic-like (which directly generate updates on the manifold itself). Among geodesic-like updates, those proposed by Wen and Yin (2013) and Jiang and Dai (2013) are the current state-of-the-art approaches. In the regime of tall-and-skinny matrices in our problem, these two are theoretically equivalent and have the same computational complexity $O(7Nn^2)$, where n is the latent dimension and $N = |\mathcal{O}|w$. By comparison, the canonical gradient projection algorithm has a slightly lower computational complexity of $O(3Nn^2)$. The exact update schemes and complexity calculations for all three methods can be found in Jiang and Dai (2013).

We compared these three update schemes by training multiple HQMMs for both synthetic HQMM and HMM datasets. As shown in the results in Figures 6a and 6b, the three methods are very similar both in terms of speed and the final solution quality for our benchmark datasets. Since the Wen-Yin update was slightly faster, especially for larger models on the synthetic HQMM data, we used it over the alternatives.

Algorithm 1 Learning HQMMs using Constrained Optimization on the Stiefel Manifold

Input: Training data $\mathbf{Y} \in \mathbb{N}^{M \times \ell}$, where M is the # of data points and ℓ is the # of observed variables in the HQMM

Hyperparameters: τ (learning rate), α: (learning rate decay), B (number of batches), E (number of epochs)

Output: $\{\mathbf{K}_i\}_{i=1}^{|\mathcal{O}|w}$

Initialize: Complex orthonormal matrix on Stiefel manifold $\boldsymbol{\kappa} \in \mathbb{C}^{|\mathcal{O}|wn \times n}$ and partition into Kraus operators $\{\mathbf{K}_i\}_{i=1}^{|\mathcal{O}|w}$, with $\mathbf{K}_i \in \mathbb{C}^{n \times n}$
for *epoch* = 1: E **do**
  Partition training data $\mathbf{Y}$ into B batches $\{\mathbf{Y}_b\}$
  for $b = 1{:}B$ **do**
    Compute gradient $\mathbf{G}_i \leftarrow \frac{\partial \mathcal{L}}{\partial \mathbf{K}_i^*}$ for batch $\mathbf{Y}_b$ and loss function $\mathcal{L}$
    Compute $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\kappa}} = \mathbf{G} \leftarrow \begin{bmatrix}\mathbf{G}_1 & \cdots & \mathbf{G}_{|\mathcal{O}|w}\end{bmatrix}^T$
    Construct $\mathbf{U} \leftarrow [\ \mathbf{G} \mid \boldsymbol{\kappa}\ ]$, $\mathbf{V} \leftarrow [\ \boldsymbol{\kappa} \mid -\mathbf{G}\ ]$
    Update $\boldsymbol{\kappa} \leftarrow \boldsymbol{\kappa} - \tau\mathbf{U}\big(\mathbb{I} + \frac{\tau}{2}\mathbf{V}^\dagger\mathbf{U}\big)^{-1}\mathbf{V}^\dagger\boldsymbol{\kappa}$
  end for
  Update learning rate $\tau = \alpha\tau$
  Re-partition $\boldsymbol{\kappa}$ into $\{\mathbf{K}_i\}$
end for
return $\{\mathbf{K}_i\}$

D Experiment on Synthetic HQMM Data

As an additional experiment on a purely quantum mechanical dataset, we compared the COSM and GS methods on data generated using the synthetic HQMM with 2 hidden states and 6 possible outputs in Srinivasan et al. (2018b). The data generation process is inspired by the well known Stern-Gerlach experiment (Gerlach and Stern, 1922) in quantum mechanics, and at least 4 hidden states are required to model it. Srinivasan et al. (2018b) demonstrated that HQMMs *learned* from such synthetic data showed in practice the same benefits that held in theory. Our goal is to verify that the COSM method performs at least as well as the GS method on a dataset well-suited to the HQMM model class.

We used the same synthetic dataset used by Srinivasan et al. (2018b), with 20 training and 10 validation sequences of length 3000. We further split up each sequence into 300 sequences and use a burn-in of 100, instead of training on 3000-length sequences with a burn-in of 1000. This reduced training time without impacting accuracy or the amount of training data processed. We trained HQMMs using the COSM approach for 60 epochs, and saved the model that yielded the highest DA score on the validation set; we used this model to evaluate on the test set of 10 sequences of length 3000 (with burn-in 1000). The results for this model are shown in Figure 7. We

see that the COSM method achieves slightly better DA compared to the GS method. We confirm that as seen in Srinivasan et al. (2018b), we need a 6−state HMM to model this 2−state HQMM.

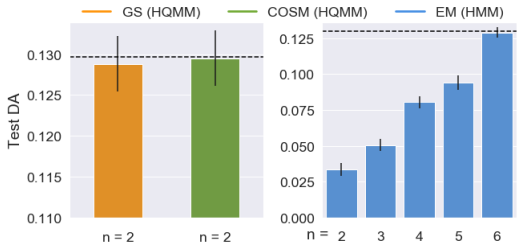


Figure 7: **Test Set Performance on the Synthetic HQMM Data**: The dashed line represents the test set performance of the true model that generated the data. The GS and COSM methods were used to learn (2,6,1)-HQMMs, while EM was used to learn HMM models with varying number of hidden states (n). A 6−state HMM model was needed to match a 2−state HQMM.

Table 1: **Hyperparameter Selection** The best performing step sizes (τ) and decay rates (α) for various COSM models. For models not listed here, the default hyperparameters (τ=0.75,α=0.92) and (τ=0.8,α=0.9) yielded the best results for the synthetic datasets and the splice dataset respectively.

Dataset	n	s	w	τ	α
Synthetic HQMM	2	6	1	0.75	0.92
Synthetic HMM	2	6	1	0.95	0.99
	4	6	6	0.95	0.96
	5	6	1	0.55	0.96
	5	6	2	0.95	0.98
	5	6	6	0.95	0.99
Splice	2	4	1	0.70	0.90
	2	4	2	0.85	0.92
	2	4	6	0.85	0.92
	4	4	1	0.90	0.92
	4	4	4	0.90	0.90
	6	4	4	0.70	0.90
	8	4	1	0.90	0.90

E Sensitivity to Initialization

The COSM algorithm begins with an initial guess of the optimal parameters $\boldsymbol{\kappa}$ and a random intial density matrix $\boldsymbol{\rho}$. By 'burning-in' a reasonable number of initial entries in sequences, we minimize the effect of randomly initializing $\boldsymbol{\rho}$. To investigate the sensitivity of COSM to initializations of $\boldsymbol{\kappa}$, we trained models on the synthetic HQMM and HMM datasets over 3 random seeds. As shown in the results in Figure 8a and 8b, COSM is sensitive to random initializations for the smallest (2,6,1) model, but the variance in DA scores quickly decrease with an increase in model size, both as a function of n and w. We observe even lower variance across different initializations for the synthetic HMM data in Figure 8b.

F Hyperparameter Selection

To facilitate a clear comparison with GS, we used the same batch size as in Srinivasan et al. (2018b), and tuned the step-size τ and decay rate α for all HQMM models. We started by manually tuning models, and identified that all models tended to converge to good solutions with the following hyperparameters: $\tau = 0.75$ and $\alpha = 0.92$ for the synthetic datasets, and $\tau = 0.8$ and $\alpha = 0.9$ for the splice dataset. We trained baseline models using these parameters, and then randomly searched for better configurations around these values.

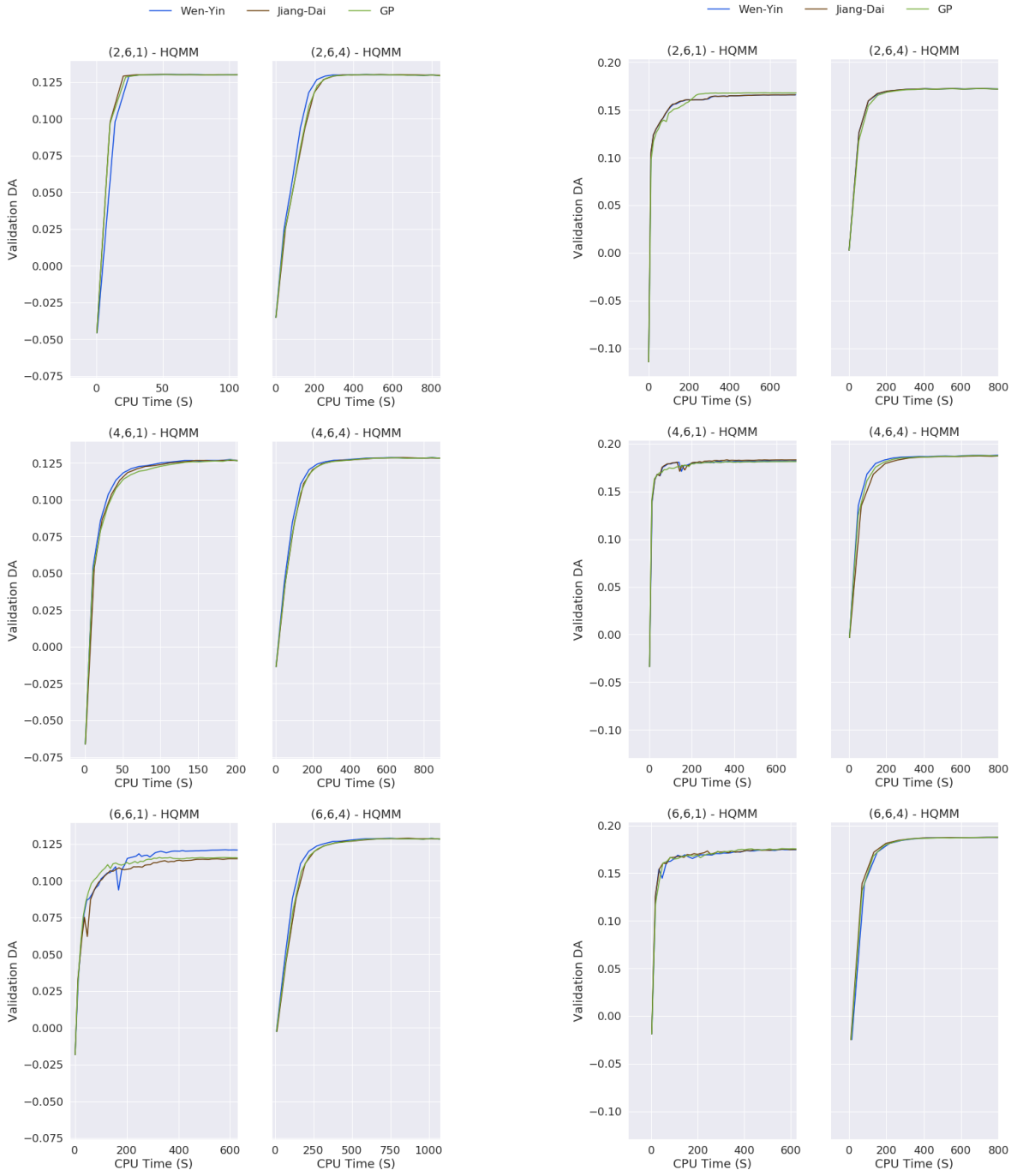
For the synthetic datasets, we fixed the batch size at 20 and randomly sampled τ between 0.55 and 0.95, and α between 0.9 and 0.99. As we wanted to explore many hyperparameter settings, we only trained on 3 random batches in every epoch. For the splice dataset, we fixed the batch size at 200 and randomly sampled τ between 0.7 and 0.9 and α between 0.88 and 0.92. Since each splice model required learning three separate HQMMs across multiple folds, we tested fewer hyperparameter settings across a smaller search space. We also trained on a single random batch every epoch across 2 folds.

Given the large number of models that we needed to evaluate, we used the Hyperband scheduling technique (Li et al., 2017) to quickly sample through many hyperparameter configurations. For each model, we began by running 3 epochs for each of the k randomly selected configurations, and removed $k/3$ of them with the lowest validation DA scores. In the next round, we ran the remaining configurations for a larger number of iterations, and again removed the bottom third of the configurations with the lowest scores. We repeated this strategy until only one configuration remained, and saved the one with the highest validation DA throughout the tuning protocol. We searched across 27 and 9 random configurations for the synthetic and the splice datasets respectively. As an example, for the synthetic datasets we trained 27 models for 3 epochs, followed by the 9 best models for 9 epochs, followed by the 3 best models for 9 epochs, and the final best model for 27 epochs. In Table 1, we report the hyperparameters obtained through Hyperband that outperformed the default configuration. For models not listed in the table, the default configuration resulted in the best performance.

All our experiments were performed on a desktop with 8 Intel Core i7-7700K 4.20 GHz CPUs, and 31.3 GB RAM. All models are trained in MATLAB, but the gradient computation happens in Python.

G Estimating Speedup

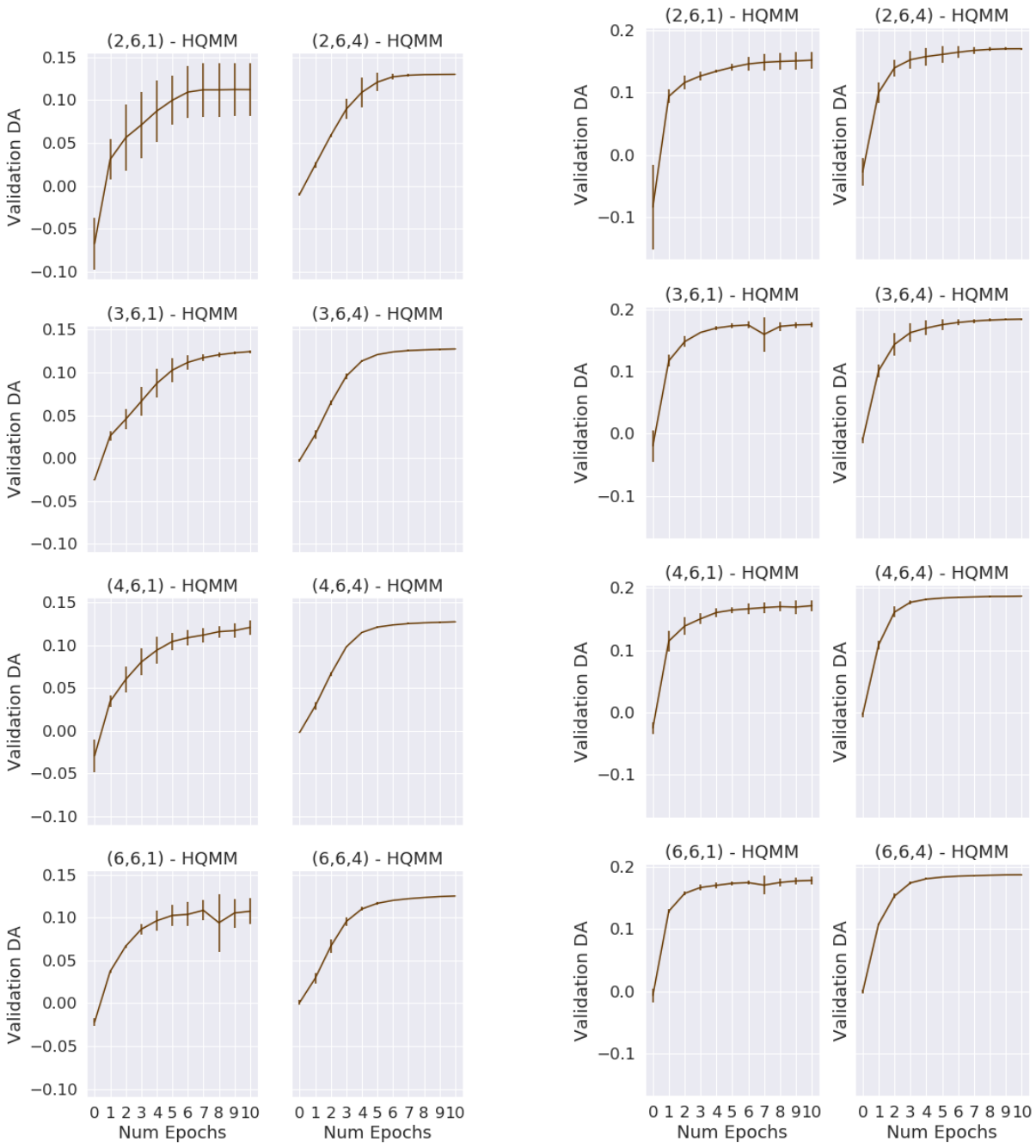
Since the GS method can take days to converge to the final solution for large models such as (6,6,6)-HQMM, it was not feasible to compute a direct speed up comparing its convergence time to COSM across most models. Thus, we estimate the speed-up offered by COSM by fitting a linear model to the DA trajectory of models learned by the GS method. Specifically, for a given HQMM model, we train both COSM and GS on the synthetic HMM data until one of them converges within a tolerance of 10^{-5} in DA scores. Since COSM always converges first, we take the DA scores achieved by GS in its last 10 steps and fit a linear model to it. We then extrapolate this linear model to estimate the time it would take for GS to reach some fraction of the solution DA reached by COSM. Note that a linear fit is an optimistic assumption of GS convergence time, meaning we are going to *understate* how much faster COSM is compared to GS. Finally, we estimate the speed up offered by COSM as the ratio of the (estimated) convergence time for GS and the actual convergence time for COSM. In Figure 9, we plot this estimated speed up with varying number of parameters (both as functions of n and w) for different solution fractions. For a solution fraction of 1, we record speedups greater than $150\times$ for the largest HQMMs trained. Furthermore, COSM offers comparable increase in speed up as parameters grow either by virtue of increasing n or w.



(a) Results for the Synthetic HQMM Data

(b) Results for the Synthetic HMM Data

Figure 6: **Alternative Schemes to Constrain Updates on the Stiefel Manifold** Validation set accuracies obtained for HQMMs trained using different update schemes. All schemes provide similar speed and accuracy, but the Wen-Yin update outperforms the others by a small margin.



(a) Synthetic HQMM Data

(b) Synthetic HMM Data

Figure 8: **COSM's Sensitivity to Random Initializations of κ** Validation set accuracies obtained across 10 epochs for HQMMs trained on 3 different random initializations. COSM is sensitive to κ initialization for the smallest models, but is fairly robust for larger models.

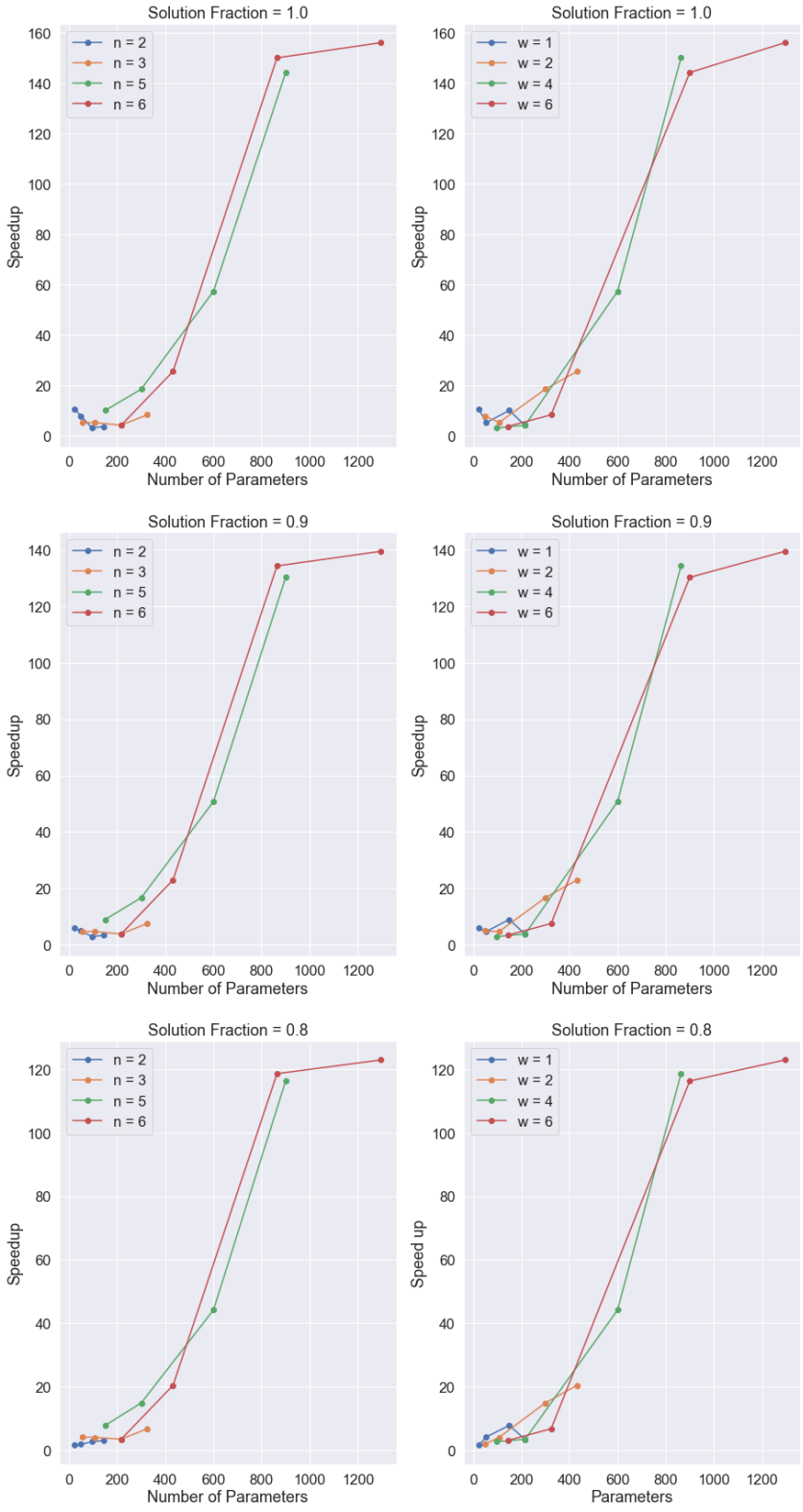


Figure 9: **Estimated Speedup of COSM over GS**: Estimated speedups of COSM over GS for various solution fractions. As seen in the plots for solution fraction of 1, GS can take more than 150 times the convergence time for COSM to reach the latter's final solution quality.